Artif Intell Rev (2007) 28:275–303
DOI 10.1007/s10462-009-9104-y

How evolutionary algorithms are applied to statistical natural language processing

Lourdes Araujo

Published online: 26 June 2009
© Springer Science+Business Media B.V. 2009

Abstract Statistical natural language processing (NLP) and evolutionary algorithms (EAs) are two very active areas of research which have been combined many times. In general, statistical models applied to deal with NLP tasks require designing specific algorithms to be trained and applied to process new texts. The development of such algorithms may be hard. This makes EAs attractive since they offer a general design, yet providing a high performance in particular conditions of application. In this article, we present a survey of many works which apply EAs to different NLP problems, including syntactic and semantic analysis, grammar induction, summaries and text generation, document clustering and machine translation. This review finishes extracting conclusions about which are the best suited problems or particular aspects within those problems to be solved with an evolutionary algorithm.

Keywords Evolutionary algorithms · Statistical natural language processing

1 Introduction

This work reviews the literature devoted to the application of any kind of evolutionary algorithm (EA) to different aspects of natural language processing (NLP). EAs (Holland 1975; Goldberg 1989) are general optimization techniques inspired on the principles of natural evolution and able to perform a guided search with a random component. This component allows them to explore the search space avoiding stagnation at local optima and accessing quickly different regions of the search space in an efficient manner. The application of EAs to NLP tasks is quite natural, as shown in Araujo (2004c). EAs do not guarantee to reach the optimum solution but an approximation, the quality of which depends on the time and memory devoted to solve the problem. However, the problems related to statistical NLP are tackled assuming a statistical model, which is itself an approximation to the real data. Therefore, it makes sense to save resources by accepting a high quality approximation.

L. Araujo (✉)
Dpto. de Lenguajes y Sistemas Informáticos, Universidad Nacional de Educación a Distancia (UNED), C/ Juan del Rosal, 16, 28040 Madrid, Spain
e-mail: lurdes@lsi.uned.es

```
evolutionary algorithm
begin
    generation ← 0
    P ← initialize_population
    F ← evaluation(P)
    while not required_fitness(F) and
            not termination_condition do
    begin
        generation ← generation + 1
        I ← individuals_selection(P, F) %for reproduction
        P ← new_generation(P, I) % genetic operators
        F ← evaluation(P)
    end
    return best(P)
end
```

Fig. 1 Structure of an evolutionary algorithm

There have been many works which have applied EAs to different NLP problems. These works have been published in different research forums: some of them have appeared in journals and proceeding of conferences devoted to evolutionary computation, others in NLP journals and conferences, and others in artificial intelligence forums, and thus they are quite dispersed. Only the work by Kool (2000) collects some applications of EAs and neural networks to some NLP problems and there have been many contributions to the subject after this report. It is the purpose of this work to collect most of them, to organize them into the different tasks included in NLP, and to extract general conclusions about most successful experiences and limitations. This study can help to develop new applications of EAs to NLP and to select the most appropriate conditions to do it.

An area closely related to NLP is information retrieval. The application of EAs to this field have been reviewed by Cordón et al. (2003) and are not included in this work.

EAs have been successfully applied in many real applications (Michalewicz 1994). An EA is an iterative technique that applies stochastic operators to a pool of potential solutions or *individuals*. A fitness function provides a value to every individual indicating its suitability to the problem. An EA usually applies a recombination operator on two solutions (*crossover*), plus a mutation one that randomly modifies the individual contents to promote diversity and thus reaching new portions of the search space not implicitly present in the previous generations. Figure 1 shows a scheme of an EA structure. The algorithm works with a population *P* of potential solutions, and has some selection process (*individuals_selection*) based on the fitness *F* of individuals (*evaluation*). The population is renewed (*new_generation*) by replacing individuals with those obtained by applying "genetic" operators to selected individuals. The production of new generations continues until resources are exhausted (*termination_condition*) or until some individual in the population is fit enough (*required_fitness*). At the end of the evolution, the solution is the best individual in the population. A commonly applied technique in evolutionary algorithms is elitism, the technique of retaining in the population for the next generation the best individuals found so far.

The most traditional EAs are genetic algorithms (GAs) (Holland 1975), which are associated to the use of a binary or integer representation. EAs which work with any kind of fixed length data structure, were introduce by Michalewicz (1994). Another approach, known as Genetic Programming (Koza 1992), instead of building an evolutionary algorithm to solve a problem, searches the space of possible programs or functions (in a particular language) for the best one. The space of programs can be regarded as a space of rooted trees, i.e., structures without a predefined size. The evaluation of an individual is based on its ability to solve a selected set of test cases.

The rest of the paper is organized in a sequence of sections, each of them devoted to a particular task of NLP: Sect. 2 is devoted to syntactic aspects of the language, Sect. 3 presents works on grammar induction, Sect. 4 describes works applying EAs to semantic NLP problems, Sect. 5 reviews works on automatic text generation, Sect. 6 to automatic text summarization, Sect. 7 is devoted to the document clustering problem, and Sect. 8 to machine translation. Each section begins with a description of the problem considered, continues with a subsection devoted to present proposals of EA application to the area, and finishes with a subsection which summarizes and draws conclusions on the work in the area. The final section is devoted to extract general conclusions and possible lines of future work in the field.

2 Syntactic analysis

In this section we consider a number of works related to language analysis which do not consider semantic aspects. Natural language is a highly complex subject, and thus, it makes sense to isolate different aspects of its processing, adopting the Chomsky's notion of the "autonomy of syntax" (Chomsky 1957). Accordingly, there has been a lot of work studying syntactic aspects specifically, such as the way in which words are grouped, the way in which such groups are nested, the word morphology and categorization or part-of-speech tagging.

Tagging aims to determine which is the correct part-of-speech (POS) tag for a particular occurrence of a word in a sentence. It is a disambiguation task since the algorithm is used to decide the tag to be assigned when there is more than one possibility. The context in which the word appears helps to decide which is its more appropriate tag, and this idea is the basis for most taggers. Automatic taggers, usually based on Hidden Markov Models, rely on statistical information to establish the probabilities of each scenario. The statistical data are extracted from previously hand-tagged texts. The Viterbi algorithm (Forney 1973), a dynamic programming one, is the most common tagging method to find the most probable tagging for a sentence according to a given Markov model, and provides a performance around 95% of accuracy on the Wall Street Journal (WSJ) data set. Other statistical models have been also proposed for tagging, such as the Brill's transformation-based part-of-speech tagger (Brill 1995). This tagger examines not only surrounding tags, but surrounding words and the morphology in the neighborhood of a particular location in the corpus. This model uses the "transformation rules" to consider the different aspects of the context. The transformation rules correspond to particular relationships among tags, words, or to particular morphology and change a particular target tag if the conditions match. A learning algorithm takes the transformation rules, which are devised a priori, and instantiates and ranks them. The solution is a ranked set of rules for tagging a corpus. This tagger has been able to reach a tagging accuracy of 97% on the WSJ data set. This section reviews some proposals of EAS applied to different models for the tagging problem.

Most natural language parsers work with a grammar extracted using supervised learning methods. This means that the grammar is extracted from a treebank, a large corpus of sentences annotated with syntactic information. Disambiguation of syntactic structures requires statistical preference. Probabilistic grammars (Charniak 1993) offer a way to establish preferences between parsings. Disambiguation techniques used in parsers for context-free grammars (CFGs) are based on probabilistic models such as probabilistic context-free grammars (PCFGs) (Charniak 1997, 2000; Collins 1997, 1999) and maximum entropy models

(Ratnaparkhi 1997). In a PCFG a weight is assigned to each rule in the grammar. In the case of corpus-driven methods, the rule probabilities are learned with the rules. The EAs used to solve the parsing problem also use supervised learning.

This section includes proposals devoted to different levels of syntactic analysis: morphological analysis (Kazakov and Manandhar 1998), annotation of the word categories in texts, i.e., part-of-speech tagging (Lankhorst 1995; Araujo 2002; Wilson and Heywood 2005) and parsing (Araujo 2004a), i.e., the search of the sentence structure with respect to a given grammar. Most works described herein use supervised learning, i.e., the parameters of the model are extracted from annotated corpus, though there is also some case that uses unsupervised learning, specifically for the tagging problem (Lankhorst 1995).

2.1 Proposals

Kazakov and Manandhar (1998) applied GAs to perform morphological analysis. Specifically, the GA is used to find the most probable word segmentation. In a second phase, the segmentation obtained is used as an input for an inductive logic programming algorithm. The final result is a logic program that can be used for segmentation of unseen words. The segmentation model adopted limits the number of word segments considered to prefixes and suffixes. The model applied is based on the Naïve Theory of Morphology (NTM) (Kazakov 1997) according to which substrings composed out of real morphemes occur in the words with a frequency higher than any other left or right substrings. For a given set of words, a segmentation can be represented by an integer vector with the segment boundary positions, each of them between zero and the word length. A segmentation defines two lexicons, prefixes and suffixes. The quality of a segmentation is estimated by its number of characters $N = P + S$ in the prefix lexicon (P) and the suffix one (S)—the smaller the number, the better the segmentation. The GA individuals are integer vectors corresponding to different segmentations, and the fitness function is $N_{\max} - N$, $N_{\max}$ being the number of characters in the word list. The proposed algorithm presents a time complexity which is linear w.r.t. the input size. Experiments were run on two set of 200 and 500 data, respectively. Results show that the GA is able to identify verb stems most of the times (coverage is among 85.0 and 98.0%, depending on the experiment). Unfortunately, the paper does not compare these results with other approaches using the same set of data, in order to further evaluate its contribution.

Lankhorst (1995) proposed to use a genetic algorithm to perform automatic word categorization in an unsupervised manner, i.e., without using annotated texts for training. The fitness function to be optimized is an estimation of an information-theoretic measure on the frequency data: the mutual information (MacKay 2003). An integer representation, in which each chromosome position corresponds to a word and represents the assigned category, is proven to perform better than a binary representation. The set of categories is inferred by the algorithm and in general, it will be different to the linguistic categorization commonly used. Experiments were carried out on sets of data taken from a 270,000 word corpus consisting of two textbooks, and the results were compared with those obtained with a greedy algorithm. The GA performs slightly better than the greedy method. However both methods have a high computational cost, because without supervision the problem is very complex. Another problem reported by the author is that the algorithm can not handle ambiguous categories adequately.

Araujo (2002) has also proposed a GA for part-of-speech tagging, in this case based on a supervised model (frequency data are extracted from manually annotated texts). The statistical model considered in this work amounts to maximizing a global measure of the probability of the set of contexts (a tag and its neighboring tags) corresponding to a given tagging of the sentence. The GA performs the search of the tagging which optimizes this measure of probability. The GA uses an integer representation in which each position corresponds to a word of the sentence and the value assigned corresponds to one of the valid part-of-speech tags for that word. One of the main contributions of this work is the facility to use different kinds of contexts, i.e., the consideration of both, tags on the right and on the left of the word being tagged. The Viterbi algorithm commonly applied to perform tagging is able to deal only with left hand side contexts, however, it has been shown (Sang 2002) that the use of contexts that include tags on the right improve the results of the Hidden Markov Models. This work shows that even small population sizes allow obtaining the correct tagging, and therefore the execution time is only slightly higher than the one employed by the Viterbi algorithm.

This work was refined later by including parallel execution of the GA for tagging, (Araujo et al. 2004; Alba et al. 2006), as well as a comparison with other heuristic search methods applied to the problem, such as CHC algorithm (a non-traditional GA with particular mechanisms to maintain diversity in the population), and simulated annealing (SA). Results obtained show that the GA is able to perform tagging with the same accuracy (96.41 for the Brown corpus and 97.32 for the Susanne corpus) as the Viterbi method (which is specific for this problem) with additional scenarios (other kinds of contexts) forbidden to other techniques. A conclusion of these works is that the use of parallelism, which is very easily implemented for GAs, allows obtaining execution times similar to those of Viterbi. The GA has been found to be better than CHC, indicating that the exploration of the search space achieved by the classical genetic operators is enough for this problem. The two evolutionary algorithms have outperformed SA. Another finding has been that an integer coding performs better than the binary one for the GA.

Wilson and Heywood (2005) have applied genetic algorithms to another POS tagging model, the Brill tagger. In this work the genetic algorithm has been used to automatically generate the rules within the Brill tagger and to rank the rules through a process of re-ordering done by the crossover operator. Each individual of the GA is composed of 378 rules (the number of rules used in the Brill's experiment with a result of 97.0% of accuracy). Each rule is a bit sequence of length 48 representing an instantiated triggering environment (rewrite rule). The collection of rules represents an entire solution to the POS tagging problem for the Wall Street Journal Corpus. The fitness function is the accuracy obtained by the 378 rules of the individual applied to some files in the WSJ corpus. The best accuracy obtained is of 89.8%, which is still far from the Brill tagger accuracy.

Let us now to considered some works where EAs are used to parse sentences. Araujo (2004a) proposed an EA to implement a probabilistic bottom-up parser which works with a population of partial parses, i.e., parses of sentence segments. The quality of the individuals is computed as a measure of its probability, which is obtained from the probability of the grammar rules and lexical tags involved in the parse. The algorithm can be view as a probabilistic implementation of a bottom-up chart parser (Araujo 2004b). The EA parser has been applied to a set of sentences extracted from the Susanne corpus (Sampson 1995) and from the Penn Treebank (Marcus et al. 1994), databases of English sentences manually annotated with syntactic information. The probabilistic grammar for parsing has also been obtained from the corpus. The results of the EA parser even improve those of a classic implementation of a chart parser for small grammars (225 rules) because the most probable parse is not always the correct one w.r.t. the corpus. The results become a bit worse when the size of the grammar is enlarged: precision drops from 99.01 using a grammar with 446 rules to 97.48 using a grammar with 795 rules, and accuracy drops from 98.20 to 97.42 for the same grammars.

The EA is able to parse sentences for which a classic implementation of a chart parser runs out of memory. However, the execution time is high for some sentences.

2.2 Conclusions

The EA proposed to perform part-of-speech tagging is valid for different models based on word contexts. The Viterbi (Forney 1973) algorithm, usually applied for tagging assuming a Hidden Markov Model, and in general any specific algorithm for a problem, will probably be more efficient than an evolutionary algorithm. However, the EA is more general and, in particular for tagging, it can be applied for contexts with tags at both sides of the word being tagged. This model is not any more a Hidden Markov model, and thus Viterbi can not be applied. Another advantage concerning the execution time is that there exists parallel versions of the EA which highly reduce the execution time without requiring additional design effort. In this way the evolutionary algorithm can reach generality and efficiency at the same time for this tagging problem and also for morphological analysis.

The EA for parsing can also be applied assuming different statistical models, which is an advantage over other exhaustive search methods. The difference will be reflected in the form of the fitness function, which in most cases can be exactly the formula or algorithm of the adopted linguistic model. Bottom-up parsing can be applied to real sentences and grammars but has a high cost. However, the EA can be very useful to perform partial parsing (parsing of sentence segments), which is needed in many applications, such as information retrieval or text summarization. Anyway, parallelism can be easily exploited in this problem. The element which increases the cost the most is the grammar size. So, improvement along this line is an open problem.

3 Grammar induction

Syntactic analysis requires a comprehensive grammar which accepts any word and syntax of the corresponding language, i.e., an exhaustive rule set. Linguists manually define finite grammars according to their linguistic knowledge. However, such grammars usually have a limited coverage since they require manually write thousands of rules and lexical entries, and thus they are not appropriate for automatic parsing of real-world sentences. Furthermore, it is not common to have the grammar available in the form of a set of rules for any given language.

Another approach amounts to constructing a corpus of sentences annotated with syntactic structures, which is called a treebank. In this case linguists are in charge of parsing real instances, and not of defining abstract rules, what will be more prone to lead to inconsistencies. Then the grammar rules can be automatically extracted from the treebank, which is called grammar induction or inference, a problem that has received much attention from many years ago (de la Higuera et al. 2005; Clark et al. 2008; van Zaanen and de la Higuera 2009). This approach, apart from facilitating the elimination of rule inconsistencies by defining an appropriate process of rule extraction, presents the advantage of providing a tool for disambiguation. The corpus provides information of statistical preference which is used by statistical models to disambiguate (Charniak 1997, 2000; Collins 1997, 1999; Ratnaparkhi 1997).

Grammar induction amounts to inferring the grammar rules of a language from positive and negative examples. The examples may or may not be annotated with the grammar structure. Obviously, annotations are a great help to the task. However, there are several reasons that make natural language grammar induction a difficult task in any case. Regular grammar inference is a hard problem since regular grammars cannot be correctly identified from positive examples alone (Gold 1967). Learning grammars from positive examples alone leads to over-generalization, i.e., the generated grammar successfully parses the training examples, but also accepts many illegal examples. The difficulties of regular grammar inference are also present in the context free grammar (CFG) case. This case gets more difficult by several decision problems which are undecidable. For example, given two CFGs, G1, and G2, there exists no algorithm that can determine whether G1 is more general than G2 (Hopcroft and Ullman 1990). A comprehensive introduction to the theoretical problems related to the induction of formal grammars can be found in (Fu and Booth 1986a,b; Parekh and Honavar 2000). These theoretical reasons make the problem difficult even in the case when we can rely on annotated corpus. Moreover, we have to take into account that the development of large enough treebanks is almost infeasible since syntactic structures must be written manually for thousands of sentences, and such treebanks are not available for most languages. In the case of stochastic grammars, i.e., grammars with probabilities added to their rules, another related problem is the estimation of such probabilities. Currently, the inside–outside algorithm (Manning and Schütze 2000) is the standard procedure for estimating the probabilities of a stochastic CFG. It is an expectation-maximization (EM) algorithm which searches for the grammar that maximizes the likelihood of the training data. However, this algorithm presents several limitations: it is very slow,[1] and is easily trapped in local maxima.

In spite of the above mentioned difficulties several statistic and heuristic approaches have been devised for grammar induction. In particular, there have been several works applying in different manners EAs to the problem. In fact, grammar induction is probably the NLP task that has received more attention from the community working on genetic and evolutionary algorithms in general. It may be due to the problem difficulty, which suggests exploring alternative methods, and also because some of the usual methods, such as the EM algorithm to train the probabilities, fail in many cases.

There are different aspects that can be considered to classify most works applying EAs to grammar induction in NLP. Some of them are focused on regular grammars (RG) (Serrano and Araujo 2005), while the others are focused on context-free grammars (CFGs). Another aspect that can be used to classify the works of this section is that some of them try to induce grammar structure (Smith and Witten 1995; Losee 1996; Korkmaz and Ucoluk 2001), while others focus on the probabilities assigned to the grammars rule (De Pauw 2003a; Serrano and Araujo 2005). A final aspect that can be used to classify these works is according to weather the proposed technique is tested on real languages (De Pauw 2003a; Serrano and Araujo 2005), or on a simple fragment of the language (Smith and Witten 1995; Losee 1996; Korkmaz and Ucoluk 2001). The attention of the GA community has concentrated not only in natural language, but also in formal languages, which provide a more accessible benchmark to study. In spite that this review is devoted to NLP applications, we will also mention here some proposals tested on formal languages which provide some insight for the natural language case.

Recall, precision and accuracy are common measures of parsing and grammar induction evaluation. *Precision* is given by the number of brackets in the parse to evaluate which match those in the correct tree; *recall* measures how many of the brackets in the correct tree are in the parse, and *accuracy* is the percentage of brackets from the parse which do not cross over the brackets in the correct parse. These measures are mentioned in some of the description below, as well as in other sections of this review.

[1] It is $\Omega m^3 n^3$, where m is the length of the sentence, and n is the number of non-terminal in the grammar.

3.1 Proposals

Smith and Witten (1995) describe a GA for the induction of a CFG in a very limited domain of natural language. They do no consider recursive rules and experiment with a small set of sentences of the type *the dog saw a cat*. This work does not assume that the words have been previously annotated with their POS tag, but tried to deduce at the same time these POS tags and the grammar. Individuals are grammars represented as LISP AND-OR s-expressions, and evolution is achieved using GA techniques for tree transformation. Individuals are selected for reproduction and mutation in proportion to its grammar size. Grammars with fewer nodes are given a higher probability of reproducing and survival than those with a larger number of nodes, according to the assumption that smaller grammars are more effective. The fitness function is based on the grammar ability to parse the current test set, which is gradually incremented. The system was able to infer the correct grammars for a small set of English sentences.

Losee (1996) has proposed a GA for learning syntactic rules and tags from scratch, i.e., without assuming a particular set of POS tags. Individuals of the GA are grammars, i.e., a set of syntactic rules. Each non-terminal symbol on the left hand side of a rule can have different right hand sides describing its direct composition, with a default value of five components. Each term has two (not necessarily different) POS assignments. The POS tags are arbitrary and have no linguistic meaning (noun, verb, etc.). The GA population is composed of three individuals, two of them being the parents and the third one the offspring produced by the parent mating. An initial grammar is randomly generated and the three individuals in the initial population are copies of this grammar. Crossover combines some rules from one parent and some from the other. Mutation produces new rules by combining fragments from the right hand side of some rules with the same left hand side as the rule it replaces, and can also produce fragments with random contents. The fitness function tries to measure the performance of the grammar on two tasks: parsing and retrieval. The measure considered for parsing is the average maximum parse length (AMPL), the largest number of terms in a parse for each sentence. The performance for retrieval and filtering is measured by the average search length (ASL) (Losee 1995). It is based on the assumption that those syntactic rules which reduce ambiguity are expected to produce a better retrieval performance. The fitness function is a combination of both functions and its computation requires parsing each sentence, which is done with a chart parser (Charniak 1993). The system has been tested on a collection of document abstracts. To simplify the grammar, only ten parts-of-speech and 20 non-terminal symbols were used. Results have shown that the GA is able to improve the quality of the initial randomly generated syntactic rules and POS labels, though the obtained grammar is quite unnatural. Unfortunately, the system performance has not been compared with other approaches.

Korkmaz and Ucoluk (2001) propose the use of genetic programming (GP) to tackle the problem. However, the straightforward application of GP to CFG induction fails. There is a high interdependency among subparts of the CFG that the usual genetic operators used in GP tend to destroy. In this work the authors use an alternative representation, transforming trees to a vectorial representation in such a manner that there is a point in an n-dimensional space assigned to each individual. This representation allows the introduction of a control module which runs a classification algorithm to determine valid and invalid chromosomes. The control module also selects to apply those genetic operations that produce offsprings from the valid chromosomes. The system has been tested considering a small subpart of the English language. The training set is composed of about 20 simple English sentences with an NP and VP structure where the VP verb can be either transitive or intransitive. Experiments show that the controlled search has a better performance compared to the straightforward application of GP.

De Pauw (2003a,b) has developed the GRAEL system, an agent-based evolutionary technique for induction and optimization of grammars for natural language. The system works with a population of agents, each of which holds a number of sentences and grammar structures. Agents exchange sentences, and if one of them is unable to reach the correct parse for the received sentence, it also receives the grammar structures required. The author has developed different instances of this environment. GRAEL-1 is devoted to optimize the probabilities of a grammar extracted from an annotated corpus. Experiments have been carried out using part of the Penn Treebank (Marcus et al. 1994) for English. The system significantly improves the parsing results over the baseline obtained using as rule probabilities their frequency in the training set. GRAEL-2 focuses on an interesting problem. Even corpus-induced or hand-written grammars are not able to cover all sentences of language and it is frequent to find sentences requiring a rule not present in the grammar. GRAEL-2 provides a mechanism to generate new rules in a guided manner. When an agent suggests a minimal correct substructure to other, the transmitted structure can suffer small mutations on different levels of the substructure, such as the deletion, addition and replacement of nodes, creating in this way new rules. The best results are obtained when GRAEL-2 is used to induce new rules, and in a second step GRAEL-1 is applied to optimize the probabilities of the new grammar. Finally, GRAEL-3 is devoted to induce unsupervised grammars. Results in this case are poorer since it is a much harder problem. The genetic operators used in this system work at the phenotype level and the fitness of an agent is based on the record of a weighted average of the F-score, which combines a measure of recall and precision, during inter-agent communication and the F-score of the agent parser on a validation set.

The work by Serrano and Araujo (2005) considered grammar inference for natural language restricted to particular components that are noun phrases (NPs). NP detection is useful for many NLP tasks, such as the identification of multiword terms, which are mainly noun phrases, or as a preprocessing step for a subsequent complete syntactic analysis, and overall, for information retrieval. This work presents a method for a *flexible* identification of basic (nonrecursive) NPs in an arbitrary text. The system generates a probabilistic finite-state automaton (FSA) able to recognize the sequences of lexical tags which form an NP. The FSA is generated with the error correcting grammatical inference (ECGI) algorithm of grammatical inference, and initial probabilities are assigned using the collected bigram probabilities.[2] The FSA probabilities provide a method for a flexible recognition of input chains, which are considered to be NPs even if they are not accepted by the FSA but are similar enough to an accepted one. Thus, the system is able to recognize NPs not present in the training examples, what has proven very advantageous for the performance of the system. The FSA probabilities, which are crucial for the flexibility in recognition, are optimized by applying an evolutionary algorithm (EA), what produces a highly robust recognizer. The EA uses both, positive and negative training examples, what contributes to improve the coverage of the system while maintaining a high precision. A comparison with other systems tested on the same set of texts has shown that the mechanism introduced for the flexible recognition of NPs clearly improves the coverage of the system, while the adjusted probabilities provided by the EA yield a high level of precision, thus providing a better overall performance.

There have also been several proposals of grammar induction devoted to extract the grammar from examples of a formal language. Some of these works are revised here because some of their novelties can be useful for grammar induction in natural language.

[2] Conditional probabilities of pairs of consecutive POS tags assigned to the words of the training text.

Wyard (1991) made one of the first proposals to applying GAs to grammar induction. The class of grammars to be inferred was restricted to context free grammars; specifically, the grammar was written in Greibach Normal Form. The formal languages considered for the experiments were the language of correctly balanced and nested brackets and the language of strings containing an equal number of a's and b's. The target grammars, which were known a priori, were used to generate sets of positive and negative samples. Each individual of the population was a list of production rules, and individuals of the initial population were generated with the same number of rules as in the target grammar. The evaluation of an individual was based on the ability of its grammar to accept positive strings and to reject negative strings. Some runs of the systems successfully arrive to a correct grammar, different from the target one, but accepting the same language. Other runs do not converge to a correct grammar. In a later work Wyard (1994) presents better results and observes that the initial population size has a great impact on them, requiring a minimum population size of 1,000 individual to avoid premature convergence, i.e., getting stuck at a local maximum. But even with this size the correct grammar was not inferred, that is why a larger size is needed for this problem according to the authors belief.

Lankhorst (1994) also proposed a genetic algorithm for CFG induction. This algorithm uses a low-level binary representation of the grammar rules. The system uses positive and negative examples, and has been tested on formal languages. An interesting contribution of this work is that it uses a fitness function which credits not only the recognition of complete strings, but also correctly analyzed substrings of the test set.

Kammeyer and Belew (1996) have also proposed a GA for the induction of stochastic CFGs. Their algorithm encodes production rules as genes of a variable length genome composed by symbols from a set appropriate to represent the grammar. This encoding allows the use of standard genetic operators. The most important novelties of this work are the introduction of intrones in the genome representation and the application of a local search operator. Intrones are "junk" regions between productions on the genotype, whose size can grow or decrease under the action of mutation and crossover. Introns provide neutral crossover points and reduce the probability of disruptive crossover. The other novelty is the tuning of the grammar probabilities before an individual evaluation, by applying the classic EM algorithm as a local search operator. Experiments on a couple of formal languages suggest that the local search operator speeds up the search.

Keller and Lutz (1997, 2005) have designed a genetic algorithm for inferring stochastic context free grammars from finite language samples. This work proposes a variant of stochastic CFG, the *biased weighted grammars* (BWG), in which each rule is associated to two parameters values: the *bias* and the *weight*. This has the advantage that the bias can be used to specify a preference for certain types of rule, using the weights to account for statistical information.

3.2 Conclusions

Most of the proposals reviewed work with a population of grammars, represented in a straightforward way in some cases, or encoded in different manners in other cases. The fitness function used to evaluate individuals is based on its ability to parse a set of training examples. Variability mainly comes from the degree of complexity of the language corresponding to the inferred grammar: real language or a restricted subpart of it. There are also differences in the GA features, such as the way in which the initial population is formed, the consideration of additional factors in the fitness, etc.

We can extract a number of conclusions observing the common points in the above described proposals as well as in their results. The more successful approaches are those devoted to tune the probabilities of the rules, since the GA avoids going into local optima, and thus can produce better results than the classic training algorithms. A limitation that the works devoted to induce the grammar structure share is that the evaluation of the candidates grammars requires parsing a number of training examples, and this makes the cost prohibitive if the tested grammar includes more than a few rules. Thus, only those EAs devoted to induce the probabilities of the rules have been applied to real languages.

However, we can also observe different positive aspects. One of them is that these systems can incorporate very easily negative examples to the training system. It is as simple as introducing some penalization for each negative example which can be parsed. Furthermore, GAs allow to take advantage of a priori knowledge that can be available for the grammar (form of the rules, size, etc.), by introducing additional factors in the individual evaluation. This knowledge can be very useful to deal with such a hard problem.

Works for grammar induction on formal languages provide several ideas whose application to natural languages are interesting to investigate, such as the hybridation of the EA with a local search such as the EM algorithm. Another interesting idea provided in these works is to credit not only the recognition of complete sentences, but also of parts of the sentence. And also to include in the fitness function other objectives which allow to specify the preference of some features of the grammar rules.

4 Semantic analysis

There have also been several proposals devoted to semantic aspects of language, i.e., interpretation and meaning, that are described in this section.

Semantics mainly refers to the meaning of words and the meaning of sentences formed by the words, as opposed to pragmatics, which refers to the meanings intended to transmit with the words. The sentence meaning depends not only on the meaning of the words, but also on rules to form word combinations, to establish word orders, etc.

One of the most common approaches to language semantics is the one based on the principle of compositionality. According to this approach the meaning of a sentence can be composed from the meaning of its parts. However this principle can not be taken in a naïve manner. The meaning of a sentence is not based only on the word composing it, but also on the ordering, grouping and relations among the words in the sentence. This means that the meaning of a sentence is partially based on its syntactic structure. Accordingly, in some cases parsing is combined with semantic aspects leading to a deep parsing, which combines syntactic and semantic aspects of the language. Examples of formalisms that belong to that category are the Head-driven phrase structure grammars (HPSGs) (Pollard and Sag 1994) which include principles and grammar rules and a lexicon with a rich structure, and the combinatory categorial grammars (CCGs) (Steedman 2000) in which grammatical categories are of two types: functors and basic categories to which the functors can be applied, and for which the C&C parser (Curran et al. 2007) has been developed. However, as far as we know, GAs have not been applied to any of them yet.

A fundamental feature of NLP is that there exist many semantically ambiguous words and sentences (with more than one meaning), and that the meaning depends on the context. This is one of the NLP problems that have received more attention within the computational linguistic community. Without a process to select the correct sense for the words in the input, the ambiguity in the lexical entries will produce an intractable amount of possible interpretations. There are two main approaches to deal with this problem. One is to perform the semantic analysis and select as correct word senses those remaining after eliminating the ill-formed semantic interpretations. The other approach consists in performing word sense disambiguation independently and a priori to the semantic analysis.

In the following, we review a work in which an EA is applied to generate a complete semantic interpretation (Rose 1999), and also a couple of works devoted to the word sense disambiguation problem (WSD) (Gelbukh et al. 2003; Decadt et al. 2004), in both cases performing the disambiguation task independently of the semantic analysis.

4.1 Proposals

Rose (1999) has applied GP to the problem of language understanding in the context of a large scale multi-lingual speech-to-speech translation system. The theory underlying this work states that the purpose of language understanding is that the listener constructs a representation of the meaning of the input sentence. Meaning representations are built from a predefined set of primitives such that meaning is encoded in the primitives themselves. The meaning of a sentence is constructed by matching a set of grammar rules against the sentence to obtain a description of the relationships between words and of how these relationships are encoded in meaning representation structures. It happens very often that the set of rules in the system are insufficient to analyze sentences that are grammatical sound. These sentences are called extra-grammatical because they are grammatical but are outside of the coverage of the system's grammar rules. However, the set of rules are usually enough to recover the meaning of most sub-expressions. Thus, the problem of robust interpretation can be thought of as the process of extracting the meaning of the sub-expressions within an extra-grammatical expression and then determining the relationships between those sub-expressions.

These ideas are implemented in the ROSE system (RObustness with Structural Evolution). The ROSE approach to the problem of extra-grammaticality is to use a robust parser to extract the meaning of subexpressions inside of a sentence. The system then uses GP to search the space of possible relationships between the meaning representation of the subexpressions. Thus, GP has here a repairing purpose. The GP stage takes as input the sub-expressions interpretations returned by the parser. The goal is to evolve a program that builds the ideal configuration out of these objects. Thus, this GP application performs two tasks in parallel: selecting the correct subset of objects, and assembling them in the correct way. The terminal set is composed of the set of objects returned by the parser. The function set contains only the COMBINE function. Objects are assembled by inserting hooks from sub-objects into a root sub-object. These sub-objects may themselves be composed of other sub-objects, and so on. The function COMBINE inserts the hook from the object that is its second argument into a hole in another object that is its first argument. The final configuration can be constructed by a program consisting only of instances of the subset of objects needed in this configuration and instances of the COMBINE function. However, not every program of this form produces a legal configuration. Each hole has restrictions about what types of objects may be inserted into it. For example, it does not make sense for an inanimate object to be the actor of an action. These restrictions are specified in a meaning representation specification that lists the full set of possible primitives, which relationships are associated with each primitive, and what types of objects can be inserted into holes corresponding to those relationships. The COMBINE function ensures that the restrictions are fulfilled. The fitness of each individual is calculated using a function that combines four goodness indicators: the number of primitive objects in the resulting structure, the number of insertions involved, a statistical score reflecting the goodness of insertions, and the percentage of the sentence that is covered by the resulting configuration. The fitness function was trained over a corpus of 48 randomly selected sentences from a corpus other than the one used in the evaluation. Each of these 48 sentences were such that repair was required for constructing a reasonable interpretation. In this corpus each sentence was coupled to its corresponding ideal meaning representation structure. To generate training data for training the repair fitness function, the repair module was run using an ideal fitness function that evaluated the goodness of hypotheses by comparing the meaning representation structure produced by the hypothesis with the ideal structure. It assigned a fitness score to the hypothesis equal to the number of primitive objects in the largest substructure shared by the produced configuration and the ideal configuration. The four scores that serve as input to the trained fitness function were extracted from each of the hypotheses constructed in each generation after the programs were ranked by the ideal fitness function. The resulting ranked lists of sets of scores were used to train a fitness function that can order the sets of scores the same way that the ideal fitness function ranked the associated hypotheses.

Typical versions of crossover and mutation in GP are used. Crossover swaps a sub-program between two parents. Mutation replaces a sub-program in the parent program by a random subprogram constructed in the same way as the initial population was generated.

ROSE's performance was evaluated in terms of efficiency and effectiveness in comparison with the two main competing approaches to robust interpretation: the maximally flexible parsing approach (Lehman 1989) and the restrictive partial parsing approach (Lavie 1995). Each approach was evaluated using the same semantic grammar with approximately 1,000 rules, with the same lexicon of approximately 3,000 lexical items, on the same previously unseen test corpus of 500 sentences. Results have shown that the ROSE system achieves a better effectiveness/efficiency trade-off than the other approaches.

In what follows we describe two applications of the evolutionary algorithm to the WSD problem.

Gelbukh et al. (2003) tackled an WSD problem taken the word senses from a dictionary and applying a genetic algorithm. The authors adopt an individual representation similar to the one used in other disambiguation task (Araujo 2002). An individual is a sequence of natural numbers varying from 1 to n_i, where n_i is the number of senses of the word w_i. Classic crossover and mutation operators are used. The fitness of an individual measures the probability of two word senses to appear in the same text fragment. Specifically, the authors use the Lesk (1986) measure. Word senses are considered to be definitions in an explanatory dictionary. Each definition is reduced to a "bag-of-words" composed of the stems of its words. The context in which a word appears is also reduced to a bag of words extracted from the definitions of all possible interpretations of the words in the context. The Lesk measure estimates the relatedness between two word sets as the number of common words in the two sets. The relatedness measure is smoothed to take the distance into account, considering it to be zero if the distance between the two words exceeds a certain threshold (text window size). Experiments showed that the proposed method, which optimizes the total word relatedness globally (within a relatively short text fragment) gives better results than existing techniques that optimize each word independently.

Decadt et al. (2004) have also applied a genetic algorithm to improve the results of a word sense disambiguation system. They use a memory-based learning approach in which all contexts of the training text containing an ambiguous word are kept in memory and only at classification time abstraction is performed by extrapolating a class from the most similar items in memory to the new one. A class is characterized by a set of features and parameters. Examples of features are keywords extracted from the training set, because of their frequent appearance in the context of a particular sense, and also from Wordnet sense definitions and the size of the context. The authors use a genetic algorithm to do joint parameter optimization and feature selection. Each individual, represented as a bit string, contains particular values for all algorithm settings. The optimization function for the GA is based on the cross-validation with the available training data. The performed experiments show that the system resulting from the GA optimization, improves the results in many cases.

4.2 Conclusions

Semantic analysis is a highly complex area, with problems which, in general, involve very large spaces (linguistic database, such as Wordnet, dictionaries, etc.). The number of EA contributions to this area is small, compared with other NLP areas, perhaps due to this inherent difficulty which can make the genetic search too expensive. However, the EAs can be very useful to solve them efficiently, whenever the problem is settled in a way that the genetic search can be appropriately guided.

The first proposal described in the section, the ROSE system (Rose 1999) is a quite ambitious one which constructs a complete representation of the meaning of a sentence, a genetic programming algorithm being in charge of repairing discrepancies among the subexpression meanings. The other proposals (Gelbukh et al. 2003; Decadt et al. 2004) included in this section tackled a well know problem in semantic interpretation, the word sense disambiguation problem. Both of them use statistical information extracted from annotated training corpus, and differ in the kind of context considered for each word. In both cases the definition of a restricted context allows the EA to perform a fine work by limiting the search space.

5 Natural language generation

Natural language generation (NLG) involves converting some kind of linguistic representation into natural language. Reiter and Dale (1997) have published an interesting introduction to the topic, where they state:

> The most common use of natural language generation technology is to create computer systems that present information to people in a representation that they find easy to comprehend. Internally, computer systems use representations which are straightforward for them to manipulate, …

The earliest generation systems relied on the use of stored text and templates to generate the new text. This approach requires preparing many predefined pieces of text in advance, to cover as much situations as possible. Templates are phrases constructed by the designer with slots which can be instantiated with pieces of text to produce new outputs. However, more complex NLG techniques can produce higher-quality texts than the use of predefined stored texts, especially in applications where there is a lot of variation in the output texts.

The task of a natural language generation system can be characterized as mapping from some input data to an output text. This complex process is usually decomposed into more precise substeps. Reiter and Dale (1997) identify the following basic activities:

– *Content determination* is the process of deciding what information should be communicated in the text.
– *Discourse planning* is the process of imposing ordering and structure over the set of messages to be conveyed.
– *Sentence aggregation* is the process of grouping messages together into sentences.

– *Lexicalization* is the process of deciding which specific words and phrases should be chosen to express the domain concepts and relations which appear in the messages.
– *Referring expression generation* is the task of selecting words or phrases to identify domain entities.
– *Linguistic realization* is the process of applying the grammar rules to produce a text which is syntactically, morphologically, and orthographically correct.

In what follows, we review some works that have approached in a more or less clear form one to more of these tasks by applying evolutionary algorithms. Karamanis and Manurung (2002) apply EAs to the discourse planning aspect, Hervás and Gervás (2005) to referring expression generation and sentence aggregation, Kim et al. (2004) to lexicalization and linguistic realization.

5.1 Proposals

Karamanis and Manurung (2002) use an evolutionary algorithm for text structuring, which can be considered part of discourse planning. These authors propose to use the principle of continuity (Grosz et al. 1995) as a predictor of the coherence of a text. The continuity principle establishes the requirement that each utterance in the discourse refers to at least one entity in the utterance that precedes it. The texts used for the experiments are short descriptions of archaeological artifacts. These texts have been analyzed into clause-sized propositions so that each clause in the text roughly corresponds to a different proposition in the database. Individuals of the evolutionary algorithm consist of an unordered set of facts or propositions. The fitness function, designed to assign a higher score to more continuous texts, is the number of continuity preservations between pairs of subsequent facts. The authors use different kinds of mutation operators: random permutation of facts, random swapping of two facts and random repositioning of a fact (removing it from its position and inserting it elsewhere, shifting the other facts accordingly). The crossover operator combines subsequences of two individuals by randomly taking a subsequence from the first parent, inserting it at a random point of the second parent, and then removing duplicate facts from the original second parent. Experiments show that the EA is able to reach the global optimum very quickly and avoids premature convergence. However, the authors point out that some of the resulting surface texts are quite incoherent.

Hervás and Gervás (2005) have also applied an EA to NLG tasks, specifically to referring expression generation and aggregation. The task of referring expression requires deciding at each occurrence how to refer to a given element. Aggregation establishes how compact the presentation of information should be in a given text, and in this work it is only applied to concepts and attributes. For instance, the system must decide between generating "The princess is blonde. She sleeps." and generating "The blonde princess sleeps." Individuals of the GA are composed of a collection of genes, each corresponding to a concept. Concepts are considered as the genes. The initial population is generated at random, using for each concept its full name or its pronoun. When using the full name, a selection of the attributes the concept has in the knowledge base is chosen. In the corresponding text, attributes will appear just before the name of the concept. The classic one point crossover is applied. The mutation operator chooses one gene at random. If the gene is a pronoun, it will change into the corresponding full name, always associated with a subset of its possible attributes. If the gene is a full name, there are two options: to change it into a pronoun, or to change the subset of attributes that appear with it. One of these two options is chosen randomly. Another operator applied is aggregation, which is only applied to full name concepts. This operator can work in two ways. If the reference to the concept appears with one or more attributes, the operator disintegrates the attributes by eliminating their mention and adding a corresponding "X is Y" sentence. If the reference to X has no attributes, the algorithm looks for an "X is Y" sentence, adds the corresponding attributes to the reference, and deletes the "X is Y" sentence. The fitness function tries to maximize the value of a set of functions which model different features of the texts generated by humans: use of correct referent, avoiding redundant attributes, avoiding reference repetition, coherence, and inclusion of concept information. The system is able to produce texts composed of a chain of short coherent sentences, though the evaluation described is limited.

Manurung (2003) has proposed a model of application of evolutionary algorithms to poetry generation. This model has been implemented as the system McGONAGALL. In this work, poetry is characterized by three features:

– Meaningfulness: a text must have associated some meaningful message.
– Grammaticality: a poem must be syntactically correct with respect to a given grammar and lexicon.
– Poeticness: a poem must present poetic features. In this case, the features considered are those referring observable aspects, such as metric and rhyme.

Individuals of the EA are possible texts with all its underlying representation, from semantics all the way down to phonetics. The fitness function measures the three requirements of meaningfulness, grammaticality, and poeticness. The grammaticality requirement is achieved by imposing that all possibly evolved solutions never violate grammatical constraints. The chosen grammar formalism is lexicalized tree adjoining grammar[3] (LTAG) (Schabes 1990).

The constraints of meaningfulness and poeticness are implemented as penalties through the evaluation function. They are the main features that will be optimized by the EA. The author proposes a number of genetic operators which are neutral of any particular grammar formalism: *add*, which adds linguistic content to a candidate solution through the application of randomly selected grammatical rules, *delete*, which removes linguistic content from a candidate solution whilst preserving syntactic well-formedness, *change*, which modifies the linguistic content of a candidate solution through the substitution of existing lexical items or syntactic structures, and *swap*, which modifies two existing linguistic structures by swapping two compatible sub-structures around. Using elitism the system produces a text that is almost metrically perfect.

Kim et al. (2004) propose the use of genetic programming to generate adaptive answers for conversational agents. Most of conversational agents respond repeatedly to users with the fixed answers stored in the reply database in advance. The goal of this work is to improve the response adaptability by responding with sentences constructed through an evolutionary process. The system extracts the keywords from the input query in a preprocessing step. The system is devoted to a specific domain and thus, keywords are defined as frequent words for that domain. The GP algorithm uses a Korean grammar in BNF (Backus Naur Form) notation, and it generates various grammar structures which are replies to the question. The replies are constructed with the query keywords and the grammar rules. The system uses an interactive evaluation in which the user is asked to put a score with five possible values, from -2 to 2. The next answer is adjusted to user's feedback, by saving those solutions with a $+2$ score. Experiments have shown that the system is able to provide more natural answers than other systems, though the evaluation of the system is quite subjective.

[3] Tree-adjoining grammar (TAG) is a grammar formalism in which the elementary unit of rewriting is the tree rather than the symbol. Whereas context-free grammars have rules for rewriting symbols as strings of other symbols, tree-adjoining grammars have rules for rewriting the nodes of trees as other trees. In a lexicalized TAG every rule is lexicalized with at least one lexical anchor, or terminal symbol.

5.2 Conclusions

In most of the described proposals the system uses some simple fitness function to evaluate a very specific aspect of the problem: a measure of the topic continuity in the case of Karamanis and Manurung (2002), the number of correct referent, redundant attributes, reference repetitions, etc. in the case of Hervás and Gervás (2005), and even the user supervision in the case of Kim et al. (2004). The search of this kind of quantitative criteria is probably the most challenging aspect of NLG, not only of the EAs application to this problem.

The most ambitious application is Manurung (2003) one. This system tries to generate poetic texts, the EAs being in charge of different aspects of the problem: meaningfulness, grammaticality and poeticness. Results in this area do not follow exactly the same criteria as other texts, because there is more freedom to construct them, and thus they can result more natural in this case. However, it is difficult to state conclusions about these results because its quality degree is quite a subjective matter.

6 Text summarization

Automated text summarization (Spärck Jones 2007) aims at extracting the main information provided by a text. This process represents a way of making information more accessible by compressing and/or combining information coming from different sources. Automatic text summarization involves some of the main problems of NLP. The process can be decomposed into three phases: analyzing the input text, transforming it into a summary representation, and synthesizing an appropriate output (Mani and Maybury 1999).

There are two main approaches to generate text summaries. The first one amounts to *extracting and rearranging* the most important words, phrases or sentences from a text. In this case, the text summarizer performs a feature analysis and estimates the possibility that a sentence belongs to the summary according to a statistical model. Assigning weights to words according to how frequently they appear in a text is a common practice in statistical text extraction. Similarly, text features are used to assign weights and to extract summary information. Different features can be considered, such as the word location in a text or the appearance in the title or first paragraphs, i.e., occurrences of words or sentences are counted and analyzed according to their frequency and position they appear in the input text.

The second approach attempts to *understand* the text and generate its summary, i.e., the systems search for a representation of the meaning of the text, identifying conceptual structures that are used to generate the summary. In this case, the source material has to be analyzed, extracting a representation that allow to synthesize the information in a new and shorter text being still intelligible.

In recent years, new tasks have been considered, such as multi-document summarization for different domains. Multi-document summarization aims at creating documents both concise and comprehensive. This task can be even harder than summarizing a single document, since the information from different documents usually lead to highest diversity.

Because text summarization is related to several NLP tasks discussed in this paper (syntactic and semantic analysis, and text generation), we have preferred to devote a separate section to this problem. The three proposals presented in this section (Yeh et al. 2002; Orăsan 2003; Andersson 2004) belong to the first approach, i.e., the goal is to construct the summary by extracting and rearranging sentences from some texts.

6.1 Proposals

Yeh et al. (2002) have developed a corpus-based approach using feature analysis for text summarization. The score function to select the relevant sentences is trained by a genetic algorithm which obtains a suitable combination of feature weights. The score assigned to a sentence to measure the probability that it belongs to the summary is calculated given the following features: *position* (important sentences are usually located at some particular positions), *positive keyword* (the more content-bearing keywords a sentence has, the more important it is), *negative keyword* (frequent keywords that are not included in the summary), *resemblance to the title* (the more overlapping keywords a sentence has with the title, the more important it is), and *centrality* (the centrality of a sentence is its similarity to other sentences, which is usually measured as the degree of common vocabulary with other sentences). The GA is in charge of obtaining an appropriate score function. The chromosome is represented as the combination of feature weights. To measure the effectiveness of an individual, the authors define the fitness as the average recall obtained when applied to the training corpus with the feature scores stored in the individual. The system has been evaluated using a data corpus composed of 100 articles about politics from New Taiwan Weekly. The use of the GA allows achieving a significant improvement on the recall.

Orăsan (2003) has used a GA to improve the quality of automatic summaries. This author applies the *continuity principle* (Grosz et al. 1995), which requires that two consecutive utterances have at least one entity in common. This principle has been previously used in the work by Karamanis and Manurung (2002), presented in the text generation section. In this work sentences are considered utterances, and two sentences are considered to have a common entity if the same head noun phrase appears in both utterances. The GA is used to select the best set of sentences to be extracted. The score of a sentence is a weighted function of several parameters: sentences which include words from the title and headers are usually important, also sentences at the beginning and the end of the documents, etc. Depending on the context in which a sentence appears in a summary, its score can be boosted or penalized. If the sentence satisfies the continuity principle with either the sentence that precedes or follows it in the summary to be produced, its score is boosted. If the continuity principle is violated the score is penalized. The weights are established through experiments. The GA uses an integer representation, taking each gene an integer value which represents the position of a sentence from the original document. The length of the chromosome is the desired length to the summary. The fitness function is the sum of the scores of the sentences indicated in the chromosome. The results obtained from the experiments show that the GA is able to improve the results of other methods applied to the same model, such as a greedy algorithm, though the achieved improvement is text dependent.

Andersson (2004) also proposes the application of a GA to extract information, but in this case the information source is not a single document, but an unrelated collection of them, such as those extracted from different web sites. In this way it is expected to select a set of data that helps the user to interpret information in new ways. The GA generates new documents from the documents resulting from the search query, and from an alphabet that scores certain words. This GA uses variable chromosome length, and the genes do not have any specific ordering. The GA continuously receives new genes as new information is available. Each gene consists of a reference to a specific sentence and a value corresponding to the sentence significance. The fitness function is calculated by adding the values of all the genes in the chromosome and normalizing them so that chromosomes with a length of five genes are favored. The function to evaluate each gene, i.e., each sentence, is computed as a linear combination of several factors: the evaluation using latent Semantic Analysis of the document the sentence belongs to, the number of query terms occurred in the sentence, the number of words from the alphabet that occur in the sentence and the reliability of the document source. Unfortunately, the system performance, that according to the author's opinion required user judgements, has not been evaluated yet.

6.2 Conclusions

Looking at the applications presented for the text summarization task, we can observe that GAs are mainly applied to refine the selection of candidate sentences to be included in the summary, or to tune the weight of the different features used in the evaluation function. A characteristic of this task is that the evaluation of the resulting summary depends on the user's preferences, or on previously collected training data. This feature is in general transmitted to the GA fitness function, what can become a bit expensive.

7 Document clustering and classification

The goal of document classification (Manning and Schütze 2000; Brücher et al. 2002) is to reduce the detail and diversity of data and the resulting information overload by grouping similar documents together. Document classification is often used to mean two types of analysis: document categorization and document clustering. Document categorization amounts to assigning documents or parts of documents in a predefined set of categories. In contrast to categorization, clustering is an unsupervised learning procedure. Clustering amounts to grouping objects into clusters such that objects from the same cluster are similar and objects from different clusters are dissimilar. Clustering analysis methods require defining some measure on the objects to be classified and a threshold value indicating whether they are similar or not.

Document clustering algorithms can be classified in partitional or hierarchical. Partitional clustering algorithms produce a simple partition of the data. Hierarchical algorithms find successive clusters using previously established clusters. Hierarchical algorithms can be agglomerative ("bottom–up") or divisive ("top–down"). Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

One of the most popular partitional algorithm for clustering is k-means, which follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. This algorithm assigns each point to the cluster whose center, called centroid, is nearest. The centroid is the average of all points associated with the documents in the cluster. k-medoid clustering is another approach for clustering similar to k-means, in which the set of representatives of the clusters is restricted to be in the set of points to classify.

The following subsection contains a work dealing with document clustering (Casillas et al. 2004), which includes a description of the way of representing the documents to apply clustering. The section also comments on some works not focused on text document clustering, but in clustering of other type of data. They have been included to show the appropriateness of the EA approach to the problem, since the different proposals can be applied to document clustering, once the documents have been represented by some numerical value. This section also includes a work devoted to document classification (del Castillo and Serrano 2004). In this case the set of categories used for the classification are discovered using an EA.

7.1 Proposals

Casillas et al. (2003, 2004) have developed a GA to perform clustering in large sets of documents. The authors approach the problem of determining at the same time the optimal number of clusters k and the best grouping of the documents into these k clusters. The approach chosen for calculating k belongs to the so called global stopping rules, which calculate the value of k for which a measure of the goodness of the partition into k clusters is optimal. This goodness measure is usually based on the within-cluster and between-cluster similarity or distance. This GA specifically uses the Calinski and Harabasz (1974) stopping rule. This method computes the minimum spanning tree (MST) of the $n \times n$ distance matrix corresponding to the n points to classify. This tree is then partitioned by removing some of its edges. For each possible partition, the within-cluster sum of squared distances to the centroids is computed. This is the fitness function used by the GA. Individuals of the GA are $n - 1$ binary vectors, where each position represents an edge of the MST. A value of zero means that the edge remains, and one means that the edge is eliminated. The number of elements with value one is $k - 1$. The algorithm uses a number of crossing points chosen at random. In order to reduce the computational cost, the authors of these works introduce sampling techniques into the problem. In the appropriate conditions, sampling allows to extract, from a reduced set of elements, conclusions valid for the whole set. In this case the sampling size is computed by fixing a sampling error and the sample is chosen at random disallowing repeated documents. The proposed clustering algorithm applies a different technique depending on the number of documents to be clustered: the Calinski and Harabasz stopping rule for small sets of documents, the GA for medium sets, and the GA with sampling for large numbers of documents. Experiments have revealed that the Calinski and Harabasz stopping rule is enough for small set of documents and a k value close to 2, but the execution time becomes unacceptable for large sets. In this case, the GA with sampling clusters the same documents in a few seconds. Furthermore, the clustering results of the GA are better according to an F-measure (a combination of recall and precision).

There have been many proposals applying EAs to the clustering problem in other domains. It is worthwhile to mention some of them because these techniques can also be applied to document clustering once the documents have been processed to be represented by a point in the space. One of the most common ways of representing documents is as a vector of terms appearing in the document. Many document clustering methods perform some preprocessing steps, such as the elimination of stop words (prepositions, determiners, etc.). Then, the document is represented as a vector of frequencies of remaining terms in the document. Distances between documents can be measured by different functions, such as the cosine of the vectors representing the documents. A complete description of these and other techniques, which are commonly used in information retrieval, can be found in Manning et al. (2008).

Sarkar et al. (1997) is one of the first proposals on applying GA to the clustering problem, to discover the optimum number k of clusters. The system uses an integer representation and the fitness function is a modification of the k-means approach. The system is able to find the correct clusters when tested on artificially generated data.

Estivill-Castro and Murray (1997) have implemented a GA to solve the k-medoid clustering problem, assuming that k is known. Individuals are integer strings and the fitness function is the sum of the distances to the cluster medoids. The authors apply random assorting recombination (RAR) (Radcliffe 1992) as crossover operator. The RAR operator is guaranteed to transmit and properly assort genes whenever this is possible. RAR takes a positive integer parameter w, places w copies of each allele present in both parents in a bag and adds to the bag one copy of each allele present in only one parent. Then, the operator repeatedly draws alleles from the bag without replacement. If the child is not fully specified at the end of this process, RAR assigns alleles to any remaining genes at random, from among the remaining legal values. The system, tested on artificial data sets, performs better than the *k*-means approach.

With the goal of reducing the computational cost, Sheng and Liu (2004) proposed the hybridation of a local search with a GA to solve the *k*-medoid clustering problem for large data sets. The local search heuristic selects *k*-medoids from the data set and minimizes the total dissimilarity within each cluster. The GA performs the global search working with individuals which are vectors of *k* integers, which represent the index of an object selected as a medoid, being *k* the number of medoids. Cluster assignment is done implicitly based on distance of the objects to the medoids. Experiments performed on gene expression data sets show that the hybrid algorithm is more efficient than other GAs proposed for the problem.

The work by del Castillo and Serrano (2004) aims at classifying incoming documents in several non-disjoint categories. The resulting system called HYCLA (HYbrid CLAssifier) learns classification models from unbalanced document samples, considering all the types of information contained in documents. The system operates in two stages, learning and integration. In the learning stage, learners apply an evolutionary technique to obtain their own feature set, and then they are trained to obtain their classification model. In the integration stage, individual learned models are evaluated on a test set, and the predictions made are combined in order to achieve the best classification of test documents. The system applies a preprocessing step in which the relevant vocabulary is extracted from the part of the document sample devoted to training. The preprocessing involves deleting stop words, stemming, and statistical selection of relevant terms. The feature space size is reduced by considering several statistical measures: information gain, mutual information, document frequency, chi square, crossover entropy, and odds-ratio. The words of all of the vocabularies are sorted by the six measurements, and only the *k* highest ranked words of each vocabulary are retained. The *k* words of each vocabulary ranked by each measurement form a view. The set of views of a vocabulary are the initial feature subsets of a learner. Views are individuals of the EA, being the chromosome length fixed to *k*. The typical size of a chromosome in text domains is about one or two thousands genes. Each gene is a word of the vocabulary. Population size matches the number of different views of a vocabulary. The fitness function of a chromosome is a measurement of the model performance computed on a test sample. The crossover operator exchanges the last third of the genes of two chromosomes to create a new offspring. The mutation operator modifies 10% of the genes from a randomly selected place in the chromosome by switching them with other possible words from the vocabulary. In HYCLA, the learners deal with a population of fixed size with six chromosomes at most. The initial population is already formed by good feature sets, and the number of generations needed to reach the final feature set is small. The HYCLA system has been evaluated using two types of digital text: scientific/technical papers and hypertext documents belonging to several categories. The EA improves the results of statistical feature selection methods, and reduces the text-domain dependence.

7.2 Conclusions

Clustering and classification have such a high computational cost that has encouraged the use of heuristic techniques, such as EAs, which have been applied with success. In the case of document clustering, the search space is usually very large, which makes this technique particularly appropriate. Nevertheless, because of the problem complexity, the best results have been obtained hybridizing the EA with other methods that reduce the search space, such as sampling (Casillas et al. 2004) or local search (Sheng and Liu 2004).

The EAs have also proven useful to select the features that a classifier must consider (del Castillo and Serrano 2004).

8 Machine translation

Machine translation (MT) (Lopez 2008) is the automatic translation of texts from one language to another one. It is a very hard problem to which many efforts have been devoted.

Clearly, the simplest approach of translating word by word does not work for many reasons: there is no one-to-one correspondence in different languages, there is lexical ambiguity and different word orders between the languages, to cite the most obvious. The next approach to MT, in order of complexity, is syntactic transfer. After parsing the source text, the parse tree is transformed into a syntactic tree in the target language. Then the translation is generated from this syntactic tree. The syntactic transfer approach solves the word order problem, but often produces an inappropriate semantics. The next level is the semantic transfer approach, which requires representing the meaning of the source sentence, and then generating the translation from the meaning. Other approaches do not rely on direct translations, but use which is called an *interlingua*. This is a knowledge representation formalism that is independent of the form in which languages express meaning.

Statistical machine translation (SMT) (Manning and Schütze 2000; Hutchins and Somers 1991) can be applied to any transfer level. Statistical translations are generated on the basis of statistical models whose parameters are obtained from the analysis of bilingual text corpora.

Bilingual, and multilingual corpora in general, can be divided into parallel and non-parallel corpora. Parallel corpora are natural language utterances and their translations with alignments between corresponding segments in different languages. Common segmentation units in parallel corpora are paragraphs and sentences. An *alignment* is the collection of links between corresponding segments.

Different approaches to automatic sentence alignment have been proposed. Length-based sentence alignment is based on statistical distance measures of the sentence lengths in terms of characters or words. Sentence alignment using lexical information selects words with similar distributions which are used as anchor for establishing sentence alignments.

In what follows we comment on several proposals applying EAs to different statistical models of MT. One of these works (Echizen-ya et al. 1996) deals with translation rules learning, others are focused in sentence translation and alignment (Otto and Rojas 2004; Rodríguez et al. 2006), while others are devoted to particular aspects related to statistical MT, such as the bilingual dictionary construction (Han 2001) or the parameter tuning of a translation system (Nabhan and Rafea 2005).

8.1 Proposals

Echizen-ya et al. (1996) propose the use of a GA to acquire translation rules. The proposal amounts to using inductive learning with genetic algorithms and is applied to the process of English–Japanese translation. Individuals are sentence translation rules consisting of an English and a Japanese sentence. A user inputs an English sentence to the system, which produces several candidate translations using the set of available translation rules. The user proofreads the resulting sentences, and this allows extracting the frequencies of correct application for the translation rules. The system determines the fitness value of the rules used in translation as the percentage of correct translations with respect to the number of times the rule has been used. At each generation new translation examples are produced by crossover and mutation. In crossover, two translation examples which have common parts are selected. The system extracts the common and different parts from all translation examples. These common and different parts are stored as new translation rules. In the experiments with sentences extracted from textbooks, precision increases about 5%.

Otto and Rojas (2004) have developed a GA to generate a translated sentence from an input sentence maximizing the translation probability. The adopted statistical model is IBM Model 4 (Brown et al. 1994), which combines the probabilities of different translation aspects: the word-by-word translation model, which is the alignment probability between words, the fertility probability of a source word, which is the probability of being aligned to a certain number of words in the target sentence, the distortion probability, which is the probability that the position of a word in the source language matches the position in the target language, and the NULL translation probability. Individuals of the GA are composed of two structures, one for alignment and other for translation. The alignment structure is a vector of a fixed length of integers which indicate the position of the target sentence. The translation structure represents the translated sentence and is a variable length string of words. The initial population is randomly generated with a greedy algorithm which attempts to ensure diversity. The GA uses different crossover operators: one point alignment crossover, lexical exchange crossover (exchanges synonymous words between the parents), and greedy lexical crossover (which selects the best translation word from the parents). The GA also uses mutation operators and specific operators which perform a local search procedure. The rate of application of the genetic operators is adapted during the evolution process. Those operators that produce better offsprings increase their probability. The system has been tested on articles from the bilingual ACM crossroads magazine. The GA results are slightly better than those obtained with other techniques such as a greedy algorithm.

Rodríguez et al. (2006) propose a particular kind of EA for searching the alignments between two sentences in a parallel corpus. The authors use an *estimation of distribution algorithm* (EDA) (Larrañaga and Lozano 2002), which differs of other EAs in the way of renewing the population. To generate new individuals, the algorithm computes the probability distribution of the genes from the current population, and then new individuals are sampled from the model thus built. The estimation/learning of the probability distribution is intractable in most cases. In practice, what is learnt is an approximation given by a factorization of a joint distribution. This work uses the Univariate Marginal Distribution Algorithm, which assumes that all the variables are marginally independent. Individuals of this EDA are vectors where each position corresponds to the alignment of the corresponding word in the input sentence. A value zero indicates that the word does not have any correspondence in the target sentence. The fitness function is the translation probability estimated by the word-based IBM statistical alignment model 4 (Brown et al. 1994), also used in the work by Otto and Rojas (2004). This model includes factors corresponding to different translation probabilities: fertility (probability of a source word to be aligned to a set of words in the target sentence), translation, head permutation, non-head permutation, null-fertility, and null-translation probabilities. The initial population is composed of randomly generated alignments between the source and the target sentences. The population size is proportional to the length of the sentences to be aligned. The probability distribution over the individuals is estimated from the 20% of the best individuals in the population. The system has been tested on different data sets showing competitive results with other algorithms such as hill-climbing. Results also show that most of the incurred translation errors were not due to the search process performed by the EDA algorithm, but to the statistical model.

The work by Nabhan and Rafea (2005) uses a GA to tune the parameters of the GIZA++ toolkit for statistical MT. GIZA++ produces alignments word alignments as part of its word based translation model, that can be used by a training process to produce some phrase-based Translation Models. The toolkit has parameters that can be used to modify the training process. These parameters are domain and data specific and should be adjusted for each corpus. Some of these parameters adopt discrete values, and are optimized by some other method, while real value parameters are optimized by a GA. Individuals are valid values of the parameter to optimize. The GA uses statistical cross entropy as the fitness function to optimize, which avoids requiring subjective judgement on the translations produced. Individuals with the lowest entropy score are selected for reproduction, and new individuals are generated with standard crossover and mutation operators. The system has been able to improve the quality translation by 11% in the experiments carried out.

Han (2001) has used a GA for the automatic construction of a bilingual dictionary in the particular case in which there is little data of the source language, which happens in minority or indigenous languages. In this case the statistical models usually applied to the problem can not be used because of the lack of data. This work relies on two assumptions: the similarity of locality and the part-of-speech (POS) distributions across the two languages. Then, the score function over a translation is defined as a linear combination of the scores contributed by each of these hypothesis. According to the locality hypothesis a good translation should map two words in a sentence of the source language into a pair of words with similar word distance in the target language sentence. The score assigned to this hypothesis is the average of the relative differences of the word distances between the source sentence and its translation. The POS distribution hypothesis says that a good translation should preserve the POS distribution. This is scored as the normalized angle between the POS distribution vectors of the sentence and its translation. The POS distribution vector of a sentence is the sum of the distribution vectors of its words, and the POS distribution of a word is the confidence that the word corresponds to the each possible POS. Individuals of the GA are translation mappings encoded in a vector. The population is initialized by randomly picking a candidate translation for each word according to the confidence distributions. The GA uses three genetic operators: crossover, mutation, and creep. Creep performs a local search to optimize the individual. At each generation, the GA selects one operator, depending on its fitness. The operator fitness value is a measure of the rewards credited to the operator responsible of the improvements over the best solution in the population. The system has been evaluated on three small corpora of different sizes, taken from the Spanish and English portions of the UN Multilingual Corpus. Though the precision and recall results of the GA improve those of a pure statistical translation, the improvements are small. As the author points out, a possible reason might be that the adopted linguistic hypotheses are insufficient.

8.2 Conclusions

EAs have been applied to different aspects of statistical MT. They have proven useful to find the correct alignment between two sentences of a parallel corpus, a task closely related to machine translation. EAs allow to apply different alignment models (based on words, phrases, etc.) by only changing the fitness function. The rest of the design is valid in every case. Thus, there are several proposals working with the same individual representation for alignment. EAs have also proven useful for tuning the parameters required in other methodologies.

However, in the case of alignment results have been less successful, since the search space is too large to be explored in a reasonable time without further restrictions.

9 General conclusions and open questions

From the above reviewed proposals we can see that EAs can be very useful for many aspects of different problems of NLP. They are specially successful when applied to specific aspects of NLP problems. For example, EAs compete with classic exhaustive search algorithms in the tagging problem, but lead to a too high cost in parsing whole sentences. Analogously, EAs have been successfully applied to alignment of sentences in a parallel corpus, but the results are very limited when applied to machine translation of sentences. In the case of grammar induction, EAs have provided high quality results for the noun phrase identification problem, but results for induction of a general grammar are not so good. The main difference comes from the fitness function. Because the search space of NLP problems is usually very large, only when the fitness function can be computed with a low cost, the execution time of the EA is viable. For specific aspects of a problem, the model to be optimized is usually simple enough to be computed with low cost, while for general problems, such as machine translation or parsing of complete sentences, the function to optimize is too complex or requires an expensive preprocessing to be applied. Besides, in many cases the statistical model for the general NLP problem is not consensually defined, but it is an open question in computational linguistics. A difference that must be taken into account in the decision process is that EAs do not guarantee the optimal solution, but an approximation whose quality depends on the devoted resources (population size, number of generations, etc.). However, when EAs are applied to statistical NLP, the underlying model is an approximation (for instance, considering that the POS tag of a word only depends on the previous one), and thus the heuristics employed in GAs can lead even to better candidates than an exhaustive search of the optimum for the statistical model.

Though EAs are in general more expensive than an algorithm specifically designed for a problem, there are several reasons that can encourage us to use them. In the first place, the computational cost of the EA can be adjusted to the available resources, i.e., we can run it with less population or less number of generations, if we need to reduce the cost. Besides, there are many NLP problems for which execution time is not critical because they do not require solution in real time. Examples of this case are corpus alignment for machine translation, grammar induction, etc.

Another motivation to apply an EA is that parallelism can be exploited very easily for them. The island model, in which the EA population is split in several processors, which run an independent EA exchanging some individuals periodically, has proven very useful and easy to implement on any computer network. Thus parallelism can compensate the extra cost of using an EA instead of a specific algorithm for the problem.

Finally, there is another important reason to choose EAs. In many cases a specific algorithm for the statistical model applied to the problem does not exist, or it is difficult to be implemented. EAs follow a general scheme which makes them easy to implement in most cases.

However, though the implementation of an EA is not difficult, obtaining high quality results requires a deep knowledge of the technique. It is necessary to make the appropriate selection of several elements, such as the individual representation, the genetic operators, the fitness function, and the running parameters. The capability of doing the correct selections requires to know the working principles of the EAs, as well as to have experience on designing and running them for different problems.

There exists a number of open source libraries which provide the main elements of an EA, such as Open Beagle[4] or GAUL,[5] just to mention two of them. However, even if we use them, we need to have a complete knowledge of the EA to be able to manage them appropriately.

In the future, as new proposals of statistical models for NLP problems are proposed, EAs could be applied as the optimization algorithm which searches for the best NLP structure according to the model. It can also be interesting to test and compare the behavior of different variants of EA: different representation (GAs, GPs, etc.), artificial ant colonies, estimation of distribution algorithms, etc. Another important challenge for the EAs which have been successfully applied to NLP is the technology transfer from the scientific papers and prototypes to professional systems.

References

Alba E, Luque G, Araujo L (2006) Natural language tagging with genetic algorithms. Inf Process Lett 100(5):173–182

Andersson L (2004) Search and creative summarization using genetic algorithms. Master Thesis. URL: citeseer.ist.psu.edu/andersson04search.html

Araujo L (2002) Part-of-speech tagging with evolutionary algorithms. In: Proceedings. of the international conference on intelligent text processing and computational linguistics (CICLing-2002), Lecture Notes in Computer Science 2276, Springer, pp 230–239

Araujo L (2004a) Genetic programming for natural language parsing. In: Proceedings of the European conference on genetic programming (EuroGP2004), Lecture Notes in Computer Science 3003, Springer, pp 230–239

Araujo L (2004b) A probabilistic chart parser implemented with an evolutionary algorithm. In: CICLing, pp 81–92

Araujo L (2004c) Symbiosis of evolutionary techniques and statistical natural language processing. IEEE Trans Evol Comput 8(1):14–27

Araujo L, Luque G, Alba E (2004) Metaheuristics for natural language tagging. In: GECCO (1), pp 889–900

Brill E (1995) Transformation–based Error–driven Learning and Natural Language Processing: A Case Study in Part–of–speech Tagging. Comput Linguist 21(4):543–565

Brown PF, Pietra SD, Pietra VJD, Mercer RL (1994) The mathematic of statistical machine translation: Parameter estimation. Comput Linguist 19(2):263–311, URL: citeseer.ist.psu.edu/brown93mathematics.html

Brücher H, Knolmayer G, Mittermayer M (2002) Document classification methods for organizing explicit knowledge. In: Proceedings of the third european conference on organizational knowledge, learning, and capabilities, Athens, Greece

Calinski T, Harabasz J (1974) A dendrite method for cluster analysis. Commun Stat 3(1):1–27

Casillas A, de Lena MTG, Martínez R (2003) Document clustering into an unknown number of clusters using a genetic algorithm. In: TSD, pp 43–49

Casillas A, de Lena MTG, Martínez R (2004) Sampling and feature selection in a genetic algorithm for document clustering. In: CICLing, pp 601–612

Charniak E (1993) Statistical language learning. MIT press, Cambridge, MA

Charniak E (1997) Statistical parsing with a context-free grammar and word statistics. In: Proceedings of the 14th national conference on artificial intelligence, AAAI Press/MIT Press, pp 598–603

Charniak E (2000) A maximum-entropy-inspired parser. In: Proceedings of the first conference on North American chapter of the association for computational linguistics, Morgan Kaufmann, San Francisco, pp 132–139

Chomsky N (1957) Syntactic structures. Mouton and Co, The Hague

Clark A, Coste F, Miclet L (eds) (2008) Grammatical inference: algorithms and applications, 9th International Colloquium, ICGI 2008, Saint-Malo, France, September 22–24, 2008, Proceedings, Lecture Notes in Computer Science, vol 5278, Springer

Collins M (1997) Three generative, lexicalised models for statistical parsing. In: Proceedings of the annual meeting of the association for computational linguistics, Association for Computational Linguistics, pp 16–23

[4] http://beagle.gel.ulaval.ca/.

[5] http://gaul.sourceforge.net.

Collins M (1999) Head-driven statistical models for natural language parsing. Ph.D. Dissertation, University of Pennsylvania

Cordón O, Herrera-Viedma E, López-Pujalte C, Luque M, Zarco C (2003) A review on the application of evolutionary computation to information retrieval. Int J Approx Reason 34(2–3):241–264

Curran JR, Clark S, Bos J (2007) Linguistically motivated large-scale nlp with c&c and boxer. In: ACL

De Pauw G (2003a) Evolutionary computing as a tool for grammar development. In: Proceedings of GECCO 2003, LNCS 2723, Springer, Berlin, pp 549–560, http://www.cnts.ua.ac.be/Publications/2003/De03f

De Pauw G (2003b) Grael: an agent-based evolutionary computing approach for natural language grammar development. In: Proceedings of the eighteenth international joint conference on artificial intelligence, Acapulco, Mexico, pp 823–828, http://www.cnts.ua.ac.be/Publications/2003/De03d

Decadt B, Hoste V, Daelemans W, van den Bosch A (2004) Gambl, genetic algorithm optimization of memory-based wsd. In: Mihalcea R, Edmonds P (eds) Proceedings of the third international workshop on the evaluation of systems for the semantic analysis of text (Senseval-3), ACL, Barcelona, pp 108–112, http://www.cnts.ua.ac.be/Publications/2004/DHDV04

de la Higuera C, Oates T, Paliouras G, van Zaanen M (eds) (2005) In: Proceedings of the IJCAI workshop on grammatical inference applications: successes and future challenges, (Held together with the 19th international joint conference on artificial intelligence), Edinburgh

del Castillo MD, Serrano JI (2004) A multistrategy approach for digital text categorization from imbalanced documents. SIGKDD Explor 6(1):70–79

Echizen-ya H, Araki K, Momouchi Y, Tochinai K (1996) Machine translation method using inductive learning with genetic algorithms. In: COLING, pp 1020–1023

Estivill-Castro V, Murray A (1997) Spatial clustering for data mining with genetic algorithms. Tech Rep FIT-TR-97-10, URL: citeseer.ist.psu.edu/estivillcastro97spatial.html

Forney GD (1973) The viterbi algorithm. Proc IEEE 61(3):268–278

Fu KS, Booth TL (1986) Grammatical inference: introduction and survey_part i. IEEE Trans Pattern Anal Mach Intell 8(3):343–359

Fu KS, Booth TL (1986) Grammatical inference: introduction and survey_part ii. IEEE Trans Pattern Anal Mach Intell 8(3):360–375

Gelbukh A, Sidorov G, Han SY (2003) Evolutionary approach to natural language word sense disambiguation through global coherence optimization. WSEAS Trans Commun 2(1):11–19

Gold EM (1967) Language identification in the limit. Inf Control 10(5):447–474

Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison Wesley, Boston

Grosz BJ, Weinstein S, Joshi AK (1995) Centering: a framework for modeling the local coherence of discourse. Comput Linguist 21(2):203–225

Han B (2001) Building a bilingual dictionary with scarce resources: a genetic algorithm approach. In: Student research workshop, the second meeting of the North American chapter of the association for computational linguistics (NAACL-2001)

Hervás R, Gervás P (2005) Applying genetic algorithms to referring expression generation. In: Tenth international conference on computer aided systems theory, EUROCAST2005

Holland JJ (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor

Hopcroft JE, Ullman JD (1990) Introduction to automata theory, languages, and computation. Addison-Wesley Longman, Boston

Hutchins W, Somers H (1991) An introduction to machine translation. Academic Press, London

Kammeyer TE, Belew RK (1996) Stochastic context-free grammar induction with a genetic algorithm using local search. In: FOGA, pp 409–436

Karamanis N, Manurung HM (2002) Stochastic text structuring using the principle of continuity. In: Proceedings of the second international natural language generation conference (INLG-02), Association for Computational Linguistics, Harriman, NY, pp 81–88, URL: citeseer.ist.psu.edu/karamanis02stochastic.html

Kazakov D (1997) Unsupervised learning of naive morphology with genetic algorithms. In: Workshop notes of the ECML/MLnet workshop on empirical learning of natural language processing tasks

Kazakov D, Manandhar S (1998) A hybrid approach to word segmentation. In: ILP'98: Proceedings of the 8th international workshop on inductive logic programming, Springer, London, pp 125–134

Keller B, Lutz R (1997) Evolving stochastic context-free grammars from examples using a minimum description length principle. In: Workshop on automata induction grammatical inference and language acquisition, ICML-97., URL: citeseer.ist.psu.edu/59698.html

Keller B, Lutz R (2005) Evolutionary induction of stochastic context free grammars. Pattern Recognition 38(9):1393–1406

Kim KM, Lim SS, Cho SB (2004) User adaptive answers generation for conversational agent using genetic programming. In: Proceedings of intelligent data engineering and automated learning (IDEAL-04), LNCS 3177, pp 813–819

Kool A (2000) Literature survey. URL: citeseer.nj.nec.com/kool99literature.html

Korkmaz EE, Ucoluk G (2001) Genetic programming for grammar induction. In: Goodman ED (ed) 2001 Genetic and evolutionary computation conference late breaking papers, San Francisco, pp 245–251, URL: citeseer.ist.psu.edu/korkmaz01genetic.html

Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge, MA

Lankhorst M (1995) Automatic word categorization with genetic algorithms. In: Proceedings of the ECAI'94, workshop on applied genetic and other evolutionary algorithms Amsterdam. Springer, URL: citeseer.ist.psu.edu/lankhorst95automatic.html

Lankhorst MM (1994) Breeding grammars. Grammatical inference with a genetic algorithm. Technical Report, Dept. of CS. University of Groningen, Groningen

Larrañaga P, Lozano J (2002) Estimation of distribution algorithms, a new tool for evolutionnary computation. Kluwer, Dordrecht

Lavie A (1995) A grammar based robust parser for spontaneous speech. Ph.D. thesis, School of Computer Science, Carnegie Mellon University

Lehman JF (1989) Adaptive parsing: self-extending natural language interfaces. Ph.D. thesis, School of Computer Science, Carnegie Mellon University

Lesk M (1986) Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In: SIGDOC '86: proceedings of the 5th annual international conference on Systems documentation, ACM Press, New York, pp 24–26, http://doi.acm.org/10.1145/318723.318728

Lopez A (2008) Statistical machine translation. ACM Comput Surv 40(3):1–49, http://doi.acm.org/10.1145/1380584.1380586

Losee RM (1995) Determining information retrieval and filtering performance without experimentation. Inf Process Manag 31(4):555–572

Losee RM (1996) Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: an empirical basis for grammatical rules. Inf Process Manag 32(2):185–197, URL: citeseer.ist.psu.edu/losee00learning.html

MacKay DJC (2003) Information theory, inference, and learning algorithms. Cambridge University Press, URL: citeseer.comp.nus.edu.sg/mackay03information.html, available from http://www.inference.phy.cam.ac.uk/mackay/itila/

Mani I, Maybury MT (1999) Advances in automatic text summarization. MIT Press, Cambridge, MA

Manning CD, Schütze H (2000) Foundations of statistical natural language processing. MIT Press, Cambridge, MA

Manning CD, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press, URL: http://www.csli.stanford.edu/~hinrich/information-retrieval-book.html

Manurung H (2003) An evolutionary algorithm approach to poetry generation. Ph.D. thesis

Marcus MP, Santorini B, Marcinkiewicz MA (1994) Building a large annotated corpus of English: the penn treebank. Comput Linguist 19(2):313–330,URL: citeseer.ist.psu.edu/marcus93building.html

Michalewicz Z (1994) Genetic algorithms + data structures = evolution programs, 2nd edn. Springer, New York

Nabhan AR, Rafea AA (2005) Tuning statistical machine translation parameters using perplexity. In: IRI, pp 338–343

Orăsan C (2003) An evolutionary approach for improving the quality of automatic summaries. In: Proceedings of the ACL 2003 workshop on multilingual summarization and question answering, Association for Computational Linguistics, Morristown, pp 37–45, http://dx.doi.org/10.3115/1119312.1119317

Otto E, Rojas MCR (2004) Towards an efficient evolutionary decoding algorithm for statistical machine translation. In: MICAI, pp 438–447

Parekh R, Honavar V (2000) Grammar inference, automata induction, and language acquisition. In: Dale R, Moisle H, Somers H (eds) Handbook of natural language processing, Marcel Dekker, New York, URL: citeseer.ist.psu.edu/rajesh00grammar.html

Pollard CJ, Sag IA (1994) Head-driven phrase structure grammar. University of Chicago Press, Chicago

Radcliffe NJ (1992) Genetic set recombination. In: FOGA, pp 203–219

Ratnaparkhi A (1997) A linear observed time statistical parser based on maximal entropy models. In: Cardie C, Weischedel R (eds) Proceedings of the second conference on empirical methods in natural language processing, Association for Computational Linguistics, Somerset, New Jersey, pp 1–10, URL: citeseer.ist.psu.edu/ratnaparkhi97linear.html

Reiter E, Dale R (1997) Building applied natural language generation systems. Nat Lang Eng 3(1):57–87, http://dx.doi.org/10.1017/S1351324997001502

Rodríguez L, Varea IG, Gámez JA (2006) Searching for alignments in smt. A novel approach based on an estimation of distribution algorithm. In: Proceedings on the workshop on statistical machine translation, Association for Computational Linguistics, New York City, pp 47–54, http://www.aclweb.org/anthology/W/W06/W06-157

Rose CP (1999) A genetic programming approach for robust language interpretation. In: Spector L, Langdon WB, O'Reilly UM, Angeline PJ (eds) Advances in genetic programming 3, MIT Press, Cambridge, chapter 4, pp 67–88, http://www.cs.bham.ac.uk/wbl/aigp3/ch04.pdf

Sampson G (1995) English for the computer. Clarendon Press, Oxford

Sang EFTK (2002) Memory-based shallow parsing. J Mach Learn Res 2:559–594

Sarkar M, Yegnanarayana B, Khemani D (1997) A clustering algorithm using an evolutionary programming-based approach. Pattern Recogn Lett 18(10):975–986

Schabes Y (1990) Mathematical and computational aspects of lexicalized grammars. PhD thesis, University of Pennsylvania, Philadelphia, available as technical report (MS-CIS-90-48, LINC LAB179) from the Department of Computer Science

Serrano JI, Araujo L (2005) Evolutionary algorithm for noun phrase detection in natural language processing. In: Congress on evolutionary computation (CEC), pp 640–647

Sheng W, Liu X (2004) A hybrid algorithm for k-medoid clustering of large data sets. In: Proceedings of the 2004 IEEE congress on evolutionary computation, IEEE Press, pp 77–82

Smith T, Witten I (1995) A genetic algorithm for the induction of natural language grammars. In: Proceedings IJCAI-95 workshop on new approaches to learning natural language, Montreal, pp 17–24

Spärck Jones K (2007) Automatic summarising: the state of the art. Inf Process Manag 43(6):1449–1481, http://dx.doi.org/10.1016/j.ipm.2007.03.009

Steedman M (2000) The syntactic process. MIT Press, Cambridge

van Zaanen M, de la Higuera C (2009) Grammatical inference and computational linguistics. In: Proceedings of the EACL 2009 workshop on computational linguistic aspects of grammatical inference, Association for Computational Linguistics, Athens, pp 1–4, http://www.aclweb.org/anthology/W09-1001

Wilson G, Heywood M (2005) Use of a genetic algorithm in brill's transformation-based part-of-speech tagger. In: GECCO '05: proceedings of the 2005 conference on genetic and evolutionary computation, ACM, New York, pp 2067–2073, http://doi.acm.org/10.1145/1068009.1068352

Wyard PJ (1991) Context free grammar induction using genetic algorithms. In: ICGA, pp 514–519

Wyard PJ (1994) Representational issues for context free grammar induction using genetic algorithms. In: ICGI '94: Proceedings of the second international colloquium on grammatical inference and applications, Springer, London, pp 222–235

Yeh JY, Ke HR, Yang WP (2002) Chinese text summarization using a trainable summarizer and latent semantic analysis. In: ICADL '02: Proceedings of the 5th international conference on Asian digital libraries, Springer, London, pp 76–87